

## See Spike Run... Run, Spike, Run

Leslie Zimmerman

*Space Telescope Science Institute, 3700 San Martin Drive, Baltimore,  
MD 21218*

**Abstract.** No, Spike is not man's best friend, but it is one the best friends that our planning and scheduling teams have. Spike (Science Planning Integrated Knowledge Environment), is an integral part of the HST planning and scheduling system. It provides long range planning information to the scheduling team; it provides bright object, timing, and orientation information to the observation planning team; and it provides graphical description of the observations and their constraints.

In testing Spike, we verify that Spike retains the knowledge of all of its old tricks, as well as the new ones. In other words, with each release of HST Spike we have to verify all or some combination of the functions Spike performs. Here, we present our strategies for regression testing the following functions of Spike: generating graphical observation descriptions, generating alerts for bright objects, relating timing and orientation link information, and long-range planning.

### Graphical User Interface (GUI)

The GUI graphically displays the constraints that are computed by Spike and stored in the description files and in the database. To test this, we exercise a subset of the menu items in the GUI and compare the display output to the corresponding values in the database or description files.

The windows that describe where the constraint suitabilities intersect ("constraint windows"), as well as the windows that show where the observation may be scheduled on the long-range plan ("plan windows"), are graphically displayed in the GUI with color-coding. The individual constraints on the observation may be viewed separately to allow one to investigate cases where the constraint suitabilities do not intersect.

In addition to the graphical display of the constraint windows and plan windows, the GUI also allows easy access to most of the Spike functions, such as loading a proposal (to calculate its observations' constraints) and running the scheduler to plan where the observations may be scheduled (which generates "plan windows").

### Bright Object Alert System (BOAS)

BOAS searches the database for bright objects associated with observations that have just been processed. BOAS then generates alerts for those bright objects.

The alert files are sent to the Contact Scientist for review. Only when the Contact Scientist has determined that the bright objects pose no health and safety threat to the instruments can the observation complete processing and be set ready for flight scheduling on the telescope.

BOAS processes all bright objects that have entered the database since the last time BOAS was run. We test BOAS by selecting several observations with bright objects in the database and setting the BOAS `time_stamp` to be before the date that the objects were entered into the database. We then compare the alert files and alert database entries to those that were generated in the previous version of BOAS. Unless there were changes made to the functionality of the bright object alert system, the files and database records should be identical.

### **Link Set Generation**

Spike translates observer specified special requirements that link observations together by timing or orient constraints into database entry commands. These commands are written to “assignment files”, which are used to populate the database with the description of the linked sets of observations.

To test this, we use a set of proposals that use a variety of the different types of timing and orient links that can be requested. We then use Spike to generate assignment files for those linked observations. If there are no changes to the functionality, then these assignment files will match those generated with the previous version of Spike.

### **Long Range Planning**

We use a specific set of proposals with a variety of the different special requirements that can be requested. These proposals are then loaded into Spike. When Spike loads the proposals, it generates its own descriptions of the observations contained within those proposals and writes those descriptions to “description files”. It also generates windows, which are the times during the plan that those observations may be performed. These “constraint windows” are an intersection of all of the constraints for an observation within the plan’s start and end times. The constraints include sun and moon avoidance, target visibility, guide star availability, and timing/orient links.

After the proposals have been successfully loaded, the Spike scheduling command is run to create a long-range plan. This scheduler uses a particular set of criteria (which are given weights by the user) to find the best places for the loaded observations to schedule with respect to one another in order to make the most efficient long range plan. Once the scheduler finishes, the “plan windows”, places where the observations may be scheduled, are written to the database.

To verify that nothing has changed in the functionality of Spike with respect to generating a long range plan, we need to satisfy ourselves that Spike has described and scheduled the observations in the same way. We do this by comparing the plan windows that we have just generated to those generated by the previous version of Spike. If the plan windows are identical, we can assume that the descriptions of the observations are also identical. Otherwise the different description would change where Spike would try to schedule the observation.